

6.10 Code 128 - barcode generation

Write a function which encodes a text using selected subset of Code 128 barcode (the same task as for MIPS project). The function should be written in Intel x86 (32 and 64 bit) assembly language. The C/C++ declaration of the function should be

```
int encode128(unsigned char *dest_bitmap,
             int bar_width,
             char *text)
```

where:

- *dest_bitmap* is a pointer to generated image (header and pixel data),
- *bar_width* - the width in pixels of narrowest bar,
- *text* – text to be encoded.

The function should return a 0 (OK) or a positive number indicating an error (e.g. barcode does not fit in bitmap of given size, text contains characters which can not be encoded).

Input

- the width in pixels of narrowest bar,
- text to be encoded.

Output

- BMP file containing the barcode image:
 - Sub format: 24 bits RGB – no compression,
 - Image size: 600x50 px,
 - Colors: bars – black, background – white.
- file name: “output.bmp”

Project versions:

1. Code Set A encoding
2. Code Set B encoding
3. Code Set C encoding
4. Encoding of any set (A, B or C).

Remarks:

1. Do not store bars and spaces patterns in coding table as character strings.
2. File I/O, memory allocation, displaying of messages should be done in C/C++ program.

References:

- [1] “**file-format-bmp**”, <https://www.daubnet.com/en/file-format-bmp>
 [2] “**Code 128**”, https://en.wikipedia.org/wiki/Code_128
 [3] Example images, <http://galera.ii.pw.edu.pl/~zsz/ecoar/images/barcodes>

4 Supplementary materials

This section contains some supplementary materials, which might be useful for your projects.

4.1 Reading data from file in Rars

The code below shows how to read data from a file:

```
#open the file
    li a7, 1024      #system call for file_open
    la a0, fname     #address of filename string
    li a1, 0         # flags: 0-read file
    ecalls           #file descriptor of opened file in a0

#save the file descriptor
    mv s1, a0

#check if file was opened correctly (file_descriptor<>-1)
    ...

#read data from file
    li a7, 63        #system call for file_read
    mv a0, s1         #move file descr from s1 to a0
    la a1, buf        #address of data buffer
    li a2, 4048       #amount to read (bytes)
    ecalls

#check how much data was actually read
    beq zero,a0, fclose    #branch if no data is read
    ...

#close the file
fclose:
    li a7, 57          #system call for file_close
    mv a0, s1          #move file descr from s1 to a0
    ecalls
```

4.2 BMP file header

The contents the header of a 24 bits RGB BMP file is shown in fig. 4.1. A hexadecimal file editor (see remark 4 below) may be used to analyze the contents of binary files. The width and height of the image is stored in 4 byte fields at 0x12 and 0x16 offsets. The offset (address) of the first pixel is stored in 32 bits *Offset of the pixel data* field at 0x0A offset. In example below pixel data starts at 0x36.

Offset	BMP marker	File size	Reserved	Offset of the pixel data	Header size
00000000	42 4D	9E D2 01 00	00 00 00 00	36 00 00 00	28 00
00000010	00 00	C8 00 00 00 Width	C7 00 00 00 Height	01 00 00 00 Planes	18 00 00 00 BPP
00000020	00 00	68 D2 00 00 Image size	13 00 00 00 X pix per meter	13 00 00 00 Y pix per meter	00 00 00 00 Colors in
00000030	00 00	00 00 Important colors	23 2E 00 00 Pixel	26 31 00 00 Pixel	28 33 00 00 Pixel
00000040	33 6C 27 34 6D 29 34 6E	29 34 6F 29 34 6F 26 33			
00000050	71 25 30 6F 25 30 6C 25	30 6B 27 31 6C 2B 35 6D			
00000060	2E 37 70 29 35 6F 25 34	6F 21 31 6D 22 32 6B 23			
00000070	32 69 26 33 6B 25 33 6D	27 35 6D 26 32 6B 25 31			
00000080	6B 26 32 6B 29 35 6D 29	34 6E 25 2F 6B 24 2F 6A			
00000090	24 2F 6B 29 33 6D 2D 37	70 27 32 6F 26 32 6B 26			

Figure 4.1. Contents of an example BMP file (all numbers in hexadecimal notation).

Remarks:

1. Little endian byte order – the first byte in a field is the least significant one.
2. Order of color components of a pixel: first byte - blue, second byte - green, third byte -red.
3. *Offset of the pixel data* field contains the offset(address) of the area where the colors of the pixels are stored.
4. Hexadecimal file editor is available at <https://hexed.it/>

References:

1. “file-format-bmp”, https://en.wikipedia.org/wiki/BMP_file_format

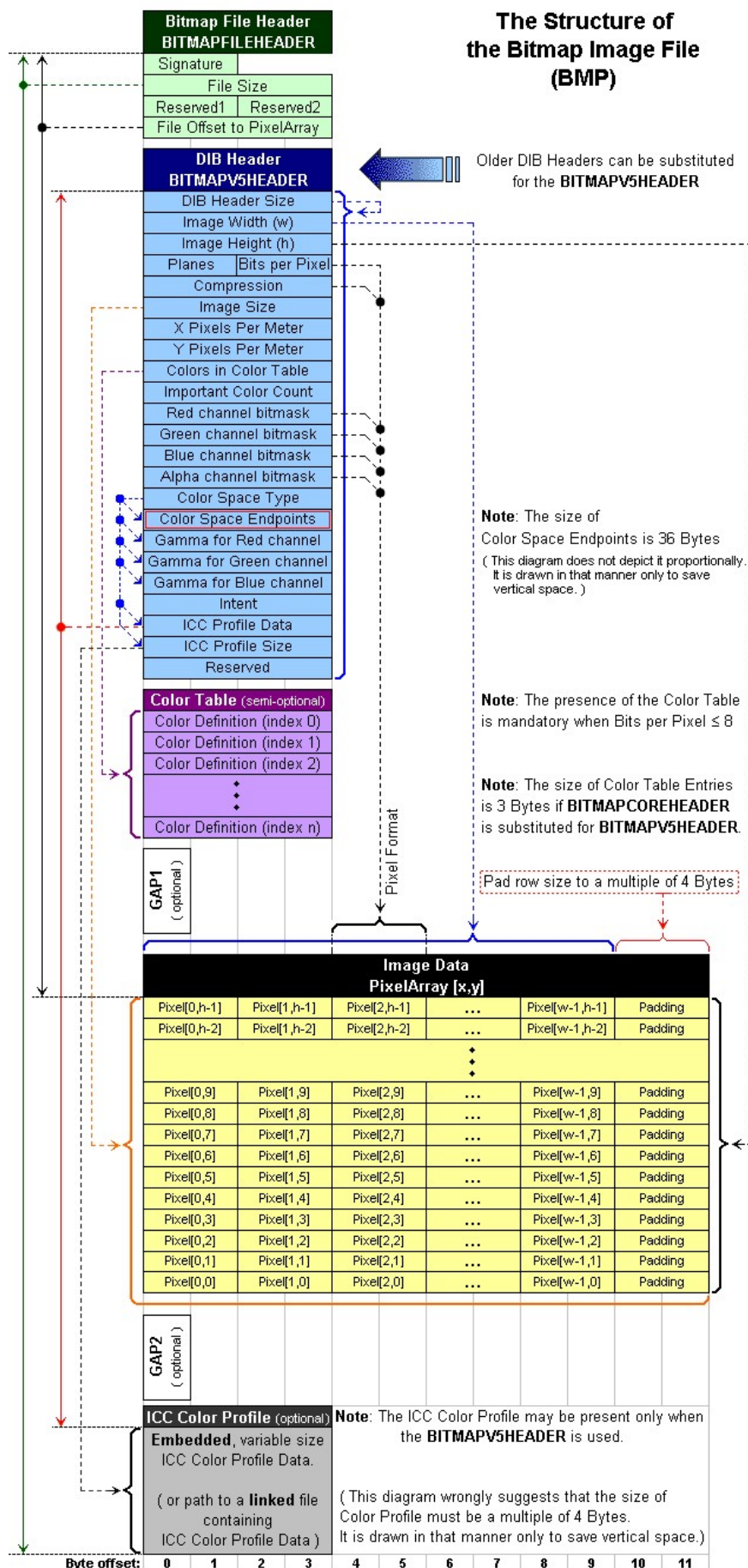


Fig. 4.2. BMP file structure [\[https://en.wikipedia.org/wiki/BMP_file_format\]](https://en.wikipedia.org/wiki/BMP_file_format)